

CS5542 - Big Data Apps and Analytics

LAB ASSIGNMENT #3

2. Spark Data Frames

Datasets:

1. **FIFA World Cup:**
<https://www.kaggle.com/abecklas/fifa-world-cup#WorldCupMatches.csv>
2. **Kickstarter Projects**
<https://www.kaggle.com/kemical/kickstarter-projects>
3. **Google-Landmarks Dataset**
<https://www.kaggle.com/google/google-landmarks-dataset>

- a. Create a Spark DataFrame using one of datasets, trying to use all different StructType.
- b. Perform 10 intuitive questions in Dataset (e.g.: pattern recognition, topic discussion, most important terms, etc.). Use your innovation to think out of box.
- c. Perform any 5 queries in Spark RDD's and Spark Data Frames. Compare the results

Data Set Considered: FIFA World Cup

Steps:

1. Creating a Spark DataFrame using different Struct type.
2. Load the data through a csv file.
3. Querying on either Data frame directly or through SQL.

Step 1: Create Spark DataFrame and Loading CSV File

The screenshot shows the PyCharm IDE with a project named 'sparkdemo'. The file explorer on the left shows a directory structure with files like 'Lab3new.py', 'WorldCupMatches.csv', and 'WorldCupPlayers.csv'. The main editor window shows the code in 'Lab3new.py'.

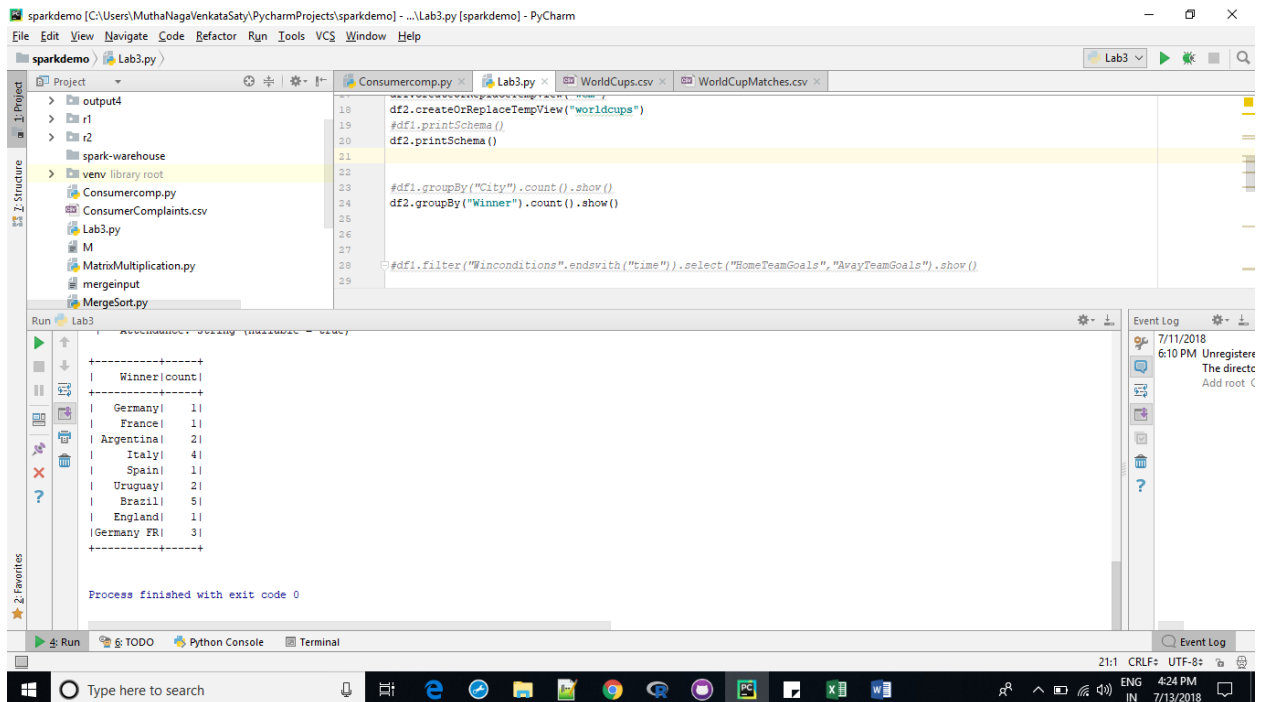
```
1 spark = SparkSession \
2     .builder \
3     .appName("Python Spark SQL basic example") \
4     .config("spark.some.config.option", "some-value") \
5     .getOrCreate()
6
7 sc = spark.sparkContext
8
9 # Load a text file and convert each line to a Row.
10 lines = sc.textFile("WorldCupMatches.txt")
11 parts = lines.map(lambda l: l.split(","))
12 # Each line is converted to a tuple.
13 matches = parts.map(lambda p: (p[0], p[1], p[2], p[3], p[4], p[5], p[6], p[7], p[8], p[9], p[10], p[11], p[12], p[13], p[14], p[15], p[16]))
14 # The schema is encoded in a string.
15 schemaString = "Year Datetime Stage Stadium City Home_Team_Name Home_Team_Goals Away_Team_Goals Away_Team_Name Win_conditions Attendance"
16 fields = (StructField(field_name, StringType(), True) for field_name in schemaString.split())
17 schema = StructType(fields)
18 # Apply the schema to the RDD
19 schemaMatches = spark.createDataFrame(matches, schema)
20 # Creates a temporary view using the DataFrame
21 schemaMatches.createOrReplaceTempView("matches")
```

The bottom of the screenshot shows the 'Run' tab with a warning message: 'NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable'. Below the warning, a table of match data is displayed.

Year	Datetime	Stage	Stadium	City	Home_Team_Name	Home_Team_Goals	Away_Team_Goals	Away_Team_Name	Win_conditions
1930	13 Jul 1930 - 15:00	Group 1	Pocitos	Montevideo	France	4	1	Mexico	
1930	13 Jul 1930 - 15:00	Group 4	Parque Central	Montevideo	USA	3	0	Belgium	
1930	14 Jul 1930 - 12:45	Group 2	Parque Central	Montevideo	Yugoslavia	2	1	Brazil	

Queries:

#Query 1: This query is directly performed on data frame itself. This query lets us know how many times each team has won the world cup till 2014.



The screenshot shows the PyCharm IDE with a project named 'sparkdemo'. The file 'Lab3.py' is open, containing the following code:

```
18 df2.createOrReplaceTempView("worldcups")
19 #df1.printSchema()
20 df2.printSchema()
21
22
23 #df1.groupBy("City").count().show()
24 df2.groupBy("Winner").count().show()
25
26
27
28 #df1.filter("Winconditions"endsWith("time")).select("HomeTeamGoals", "AwayTeamGoals").show()
29
```

The Run console shows the output of the queries:

```
Winner|count|
-----+-----+
| Germany| 1|
| France| 1|
| Argentina| 2|
| Italy| 4|
| Spain| 1|
| Uruguay| 2|
| Brazil| 5|
| England| 1|
| Germany FR| 3|
-----+-----+
```

Process finished with exit code 0

The Event Log shows the following events:

- 7/11/2018 6:10 PM Unregister The direct Add root C

#Query 2: This query gives the total number of goals scored by each team with the team scoring more goals as the first.

The screenshot shows the PyCharm IDE with a project named 'sparkdemo'. The file explorer on the left shows a directory structure with files like 'spark-warehouse', 'venv', 'library root', 'Consumercomp.py', 'ConsumerComplaints.csv', 'Lab3.py', 'M', and 'MatrixMultiplication.py'. The main editor window displays a Python script in 'Lab3.py' with the following code:

```
54 #Total number of goals scored by a country
55 query10 = spark.sql("SELECT HomeTeamName AS Team, SUM(HomeTeamGoals) AS TOTAL_GOALS from t1 GROUP BY HomeTeamName ORDER BY 2 DESC")
56 query10.show()
57
58
59
60
61
```

The 'Run' tab at the bottom shows the execution results of the query, displaying a table with the following data:

Team	TOTAL_GOALS
Brazil	180.0
Argentina	111.0
Italy	99.0
Germany FR	99.0
Hungary	73.0
Germany	69.0
France	68.0
Uruguay	62.0
England	54.0
Sweden	53.0
Netherlands	51.0
Spain	50.0
Soviet Union	43.0
Yugoslavia	42.0
Portugal	36.0
Austria	31.0
Czechoslovakia	27.0
Belgium	27.0
Poland	27.0
Chile	25.0

The bottom status bar shows the system clock as 5:43 PM on 7/16/2018.

#Query3: This query tells the average number of goals scored by team.

The screenshot shows the PyCharm IDE with the same project 'sparkdemo'. The file explorer on the left shows a directory structure with files like 'sparkdemo', 'mergeoutput', 'output3', 'output4', 'r1', 'r2', 'spark-warehouse', and 'venv'. The main editor window displays a Python script in 'Lab3.py' with the following code:

```
54 # Max_Attendance.show()
55
56 Avg_goals = spark.sql("SELECT HomeTeamName AS Team_Name, ROUND(AVG(HomeTeamGoals),0) AS Average_Goals FROM wcm GROUP BY HomeTeamName")
57 Avg_goals.show()
58
59
60
61
```

The 'Run' tab at the bottom shows the execution results of the query, displaying a table with the following data:

Team_Name	Average_Goals
Paraguay	1.0
Russia	3.0
Senegal	2.0
Sweden	2.0
IR Iran	0.0
Turkey	5.0
Zaire	0.0
Iraq	1.0
Germany	2.0
France	2.0
Greece	1.0
Algeria	1.0
Togo	0.0
Slovakia	2.0
null	null
Argentina	2.0
Wales	2.0
Belgium	2.0
Angola	0.0
Ecuador	1.0

The bottom status bar shows the system clock as 5:59 PM on 7/16/2018.

#Query 4:

This query is bit about the pattern recognition. I found the matches where the matches have taken extra time to win the game.

The screenshot shows the PyCharm IDE with a Spark SQL query in the editor and its results in the Run console. The query is designed to find matches where the winning team took extra time to win, based on the 'Winconditions' field.

```
#g4 = spark.sql("select wcm.HomeTeamGoals,wcm.AwayTeamGoals from wcm where wcm.Winconditions LIKE 'extra'.show()")
#Pattern_reg = spark.sql("SELECT substring(W.Winconditions,' ',1),W.MatchID as Winner from WCM W FULL JOIN(SELECT * from wcm v WHERE v.Winner = W.MatchID) ON v.Winner = W.MatchID")
Pattern_reg = spark.sql("SELECT * from wcm w WHERE w.Winconditions LIKE '%extra%'").show()
```

The Run console displays a table with the following columns: Year, Datetime, Stage, Stadium, City, HomeTeamName, HomeTeamGoals, AwayTeamGoals, AwayTeamName, and Winconditions. The table lists various World Cup matches from 1934 to 1998, including preliminary rounds, group stages, quarter-finals, semi-finals, and finals. The 'Winconditions' column shows the result of each match, such as 'France/Austria win after e.' or 'Italy/Italy win after e.'

Year	Datetime	Stage	Stadium	City	HomeTeamName	HomeTeamGoals	AwayTeamGoals	AwayTeamName	Winconditions
1934	127 May 1934 - 16:30	Preliminary round	Stadio Benito Mus...	Turin	Austria	3	2	France	Austria win after e.
1934	110 Jun 1934 - 17:30	Final	Nazionale PNF	Rome	Italy	2	1	Czechoslovakia	Italy win after e.
1938	105 Jun 1938 - 17:00	First round	Stade Vélodrome	Marseille	Italy	2	1	Norway	Italy win after e.
1938	105 Jun 1938 - 17:30	First round	Stade de la Meinau	Strasbourg	Brazil	6	5	Poland	Brazil win after e.
1938	105 Jun 1938 - 18:30	First round	Cavee Verte	Le Havre	Czechoslovakia	3	0	Netherlands	Czechoslovakia win after e.
1954	130 Jun 1954 - 18:00	Semi-finals	La Pontaise	Lausanne	Hungary	4	2	Uruguay	Hungary win after e.
1958	117 Jun 1958 - 19:00	Group 1	Malmö Stadion	Malmö	Northern Ireland	2	1	Czechoslovakia	Northern Ireland win after e.
1966	130 Jul 1966 - 15:00	Final	Wembley Stadium	London	England	4	2	Germany FR	England win after e.
1970	14 Jun 1970 - 12:00	Quarter-finals	Nou Camp - Estadi...	Leon	Germany FR	3	2	England	Germany FR win after e.
1970	14 Jun 1970 - 12:00	Quarter-finals	Estadio Azteca	Mexico City	Uruguay	1	0	Soviet Union	Uruguay win after e.
1970	17 June 1970 - 16:00	Semi-finals	Estadio Azteca	Mexico City	Italy	4	3	Germany FR	Italy win after e.
1978	25 Jun 1978 - 15:00	Final	El Monumental - E...	Buenos Aires	Argentina	3	1	Netherlands	Argentina win after e.
1986	15 Jun 1986 - 16:00	Round of 16	Nou Camp - Estadi...	Leon	Soviet Union	3	4	Belgium	Belgium win after e.
1986	28 Jun 1986 - 12:00	Match for third p...	Cuauhtemoc	Puebla	France	4	2	Belgium	France win after e.
1990	23 Jun 1990 - 17:00	Round of 16	San Paolo	Naples	Cameroon	2	1	Colombia	Cameroon win after e.
1990	26 Jun 1990 - 17:00	Round of 16	Marc Antonio Bent...	Verona	Spain	1	2	Yugoslavia	Yugoslavia win after e.
1990	26 Jun 1990 - 21:00	Round of 16	Renato Dall Ara	Bologna	England	1	0	Belgium	England win after e.
1990	01 Jul 1990 - 21:00	Quarter-finals	San Paolo	Naples	England	3	2	Cameroon	England win after e.
1994	05 Jul 1994 - 13:00	Round of 16	Foxboro Stadium	Boston	Nigeria	1	2	Italy	Italy win after e.
1998	28 Jun 1998 - 16:30	Round of 16	Stade Felix Bollaert	Lens	France	1	0	Paraguay	France win after e.

#Query 5: In this query, I tried to find where the Home Team leads the Away Team by halftime but the Away Team wins the match.

The screenshot shows a PyCharm IDE with a Spark SQL query in `Lab3.py`. The query is designed to find matches where the home team leads at halftime but the away team wins the match. The results are displayed in a table with columns: Stage, Stadium, City, HomeTeamName, HomeTeamGoals, AwayTeamGoals, AwayTeamName, Winconditions, Attendance, HalftimeHomeGoals, and HalftimeAwayGoals.

```

17 # homeadv.show()
18
19 q1 = spark.sql("""select * from wcm where wcm.HalftimeHomeGoals>wcm.HalftimeAwayGoals""")
20 q1.createOrReplaceTempView("t1")
21 #q1.show()
22 q2 = spark.sql("""select * from wcm where wcm.AwayTeamGoals>wcm.HomeTeamGoals""")
23 q2.createOrReplaceTempView("t2")
24 #q2.show()
25 q3=spark.sql("""select * from t1,t2 where t1.MatchID=t2.MatchID""")
26 q3.show()
27

```

Stage	Stadium	City	HomeTeamName	HomeTeamGoals	AwayTeamGoals	AwayTeamName	Winconditions	Attendance	HalftimeHomeGoals	HalftimeAwayGoals
roup C	Luigi Ferraris	Genoa	Sweden	1	2	Costa Rica		30223	1	0
roup E	Stade de Gerland	Lyon	Korea Republic	1	3	Mexico		39100	1	0
roup B	Jeju World Cup St...	Jeju	Slovenia	1	3	Paraguay		30176	1	0
roup D	Estadio Castelao	Fortaleza	Uruguay	1	3	Costa Rica		58679	1	0

#Query 6: In this query, I tried to find the max attendance for a stadium

The screenshot shows a PyCharm IDE with a Spark SQL query in `Lab3.py`. The query is designed to find the maximum attendance for each stadium. The results are displayed in a table with columns: Stadium and Max_Attendance.

```

83 # query10.show()
84
85 # left_outer_join = spark.sql("""SELECT A.City,A.HomeTeamName,A.AwayTeamName,A.Attendance,B.Max_Attendance from t1 A LEFT OUTER JOIN (SE
86 # left_outer_join.show()
87
88 # test = spark.sql("""SELECT Stadium,MAX(Attendance) AS Max_Attendance FROM t2 B GROUP BY Stadium""").show()
89 df3.createOrReplaceTempView("t3")
90
91 df10J = spark.sql("""select DISTINCT A.Stadium, B.Max_Attendance from t1 A left outer join ( select Stadium, max(Attendance) Max_Att
92

```

Stadium	Max_Attendance
Ilha do Retiro	85011
St. Jakob	58000
Parkstadion	68348
Durivel de Brito	95111
Malmo Stadion	61961
Seoul World Cup S...	65256
Victor Bouquey	15000
Stadio delle Alpi	62628
Ellis Park Stadium	55686
Estadio Corregidora	38500
FIFA World Cup St...	48000
Hardturni	32000
Camp Nou	95000
Kobe Wing Stadium	40440
Bombonera - Estad...	24000
Foxboro Stadium	54456
Santiago Bernabeu	90089
Giuseppe Meazza	74765

The screenshot shows the PyCharm IDE interface. The top toolbar includes icons for Run, Debug, and other development actions. The main editor window displays a Python script named `Lab3new.py` containing Spark SQL code. The script reads a CSV file, creates an RDD, applies a schema, and executes a SQL query to filter matches where the home team scored 4 or more goals. The console at the bottom shows the output of the query, displaying a table with columns `COUNTRY` and `No_of_Times`. The right sidebar contains the Event Log, which shows messages about the IDE and plugin updates.

```
# Each line is converted to a tuple.
matches = parts.map(lambda p: (p[0], p[1], p[2], p[3], p[4], p[5], p[6], p[7], p[8], p[9], p[10], p[11], p[12], p[13], p[14], p[15], p[16]))
# The schema is encoded in a string.
schemaString = "Year Datetime Stage Stadium City Home_Team_Name Home_Team_Goals Away_Team_Goals Away_Team_Name Win_conditions Attendance"
fields = [StructField(field_name, StringType(), True) for field_name in schemaString.split(",")]
schema = StructType(fields)
# Apply the schema to the RDD
schemaMatches = spark.createDataFrame(matches, schema)
# Creates a temporary view using the DataFrame
schemaMatches.createOrReplaceTempView("matches")
query5 = spark.sql("SELECT Home_Team_Name AS COUNTRY,COUNT(Home_Team_Goals) AS No_of_Times FROM matches where Home_Team_Goals >=4 GROUP BY Home_Team_Name")
```

COUNTRY\No_of_Times	
Brazil	18
Germany FR	11
Hungary	10
France	8
Argentina	8
Uruguay	6
Germany	5
Italy	5
Yugoslavia	5
Soviet Union	3
Netherlands	3
Portugal	2
Sweden	2
England	2
Austria	2
Spain	2

The screenshot shows the PyCharm IDE interface. The top toolbar includes buttons for File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The main editor window displays the code for 'Lab3new.py'. The code defines a schema for a DataFrame, creates the DataFrame from a CSV file, and executes two SQL queries. The first query filters for matches where the home team goals are greater than or equal to 4. The second query filters for matches where the home team goals are greater than 3 and less than or equal to 10. The bottom pane shows the 'Run' output, which displays a table of match data.

	Stage	Stadium	City/Home_Team_Name
Group 1	Pocitos	Montevideo	France
Group 4	Parque Central	Montevideo	USA
Group 3	Pocitos	Montevideo	Romania
Group 1	Parque Central	Montevideo	Chile
Group 2	Parque Central	Montevideo	Yugoslavia
Group 4	Parque Central	Montevideo	USA
Group 1	Estadio Centenario	Montevideo	Argentina
Group 2	Estadio Centenario	Montevideo	Brazil
Group 3	Estadio Centenario	Montevideo	Uruguay
Group 1	Estadio Centenario	Montevideo	Argentina
Semi-finals	Estadio Centenario	Montevideo	Argentina
Semi-finals	Estadio Centenario	Montevideo	Uruguay
Final	Estadio Centenario	Montevideo	Uruguay
Preliminary round	Stadio Benito Mus...	Turin	Austria
Preliminary round	Giorgio Ascarelli	Naples	Hungary

#Query 9: I tried to find the max goals between a range of years

sparkdemo [C:\Users\MuthaNagaVenkataSaty\PycharmProjects\sparkdemo] - ...Lab3new.py [sparkdemo] - PyCharm

File Edit View Navigate Code Refactor Run Tools VCS Window Help

sparkdemo Lab3new.py

Project

- output4
- r1
- r2
- spark-warehouse
- venv library root
- ConsumerComp.py
- ConsumerComplaints.csv
- ks-projects-201612.csv
- Lab3.py
- Lab3new.py
- M
- MatrixMultiplication.py

Run: Lab3new Lab3

```
22 schemaMatches.createOrReplaceTempView("matches")
23 #query5 = spark.sql("SELECT Home_Team_Name AS COUNTRY,COUNT(Home_Team_Goals) AS No_of_Times FROM matches where Home_Team_Goals >=4 GROU
24
25
26 #query1 = spark.sql("SELECT Stage,Stadium,City,Home_Team_Name FROM matches WHERE Home_Team_Goals >= 3 AND Home_Team_Goals <= 10").show(
27
28 q9 = spark.sql("SELECT Home_Team_Name, MAX(Home_Team_Goals) from matches where Year BETWEEN 2000 AND 2010 GROUP BY Home_Team_Name ORDER
```

Germany| 8|
Portugal| 7|
Spain| 4|
Brazil| 4|
Argentina| 4|
Senegal| 3|
Poland| 3|
Belgium| 3|
Turkey| 3|
Australia| 3|
USA| 3|
Mexico| 3|
Slovakia| 3|

only showing top 20 rows

Process finished with exit code 0

Event Log

7/16/2018
9:40 AM IDE and Plugin U
9:40 AM Unregistered
The directory C:\
[Add root](#) [Config](#)

Unregistered VCS root detected: The directory C:\Users\MuthaNagaVenkataSaty is under Git, but is not registered in the Settings. // Add root Configure Ignore (today 9:40 AM)

25:1 CRLF: UTF-8: ENG 11:32 AM 7/16/2018

#Query 10: I tried to find the teams and the years they won using collect_set function

sparkdemo [C:\Users\MuthaNagaVenkataSaty\PycharmProjects\sparkdemo] - ...Lab3.py [sparkdemo] - PyCharm

File Edit View Navigate Code Refactor Run Tools VCS Window Help

sparkdemo > Lab3.py

Project

- output4
- r1
- r2
- spark-warehouse
- venv library root
- Consumercomp.py
- ConsumerComplaints.csv
- ks-projects-201612.csv
- Lab3.py
- Lab3new.py
- M
- MatrixMultiplication.py

Run: Lab3new Lab3

```
5 spark = SparkSession \
6     .builder \
7     .appName("Python Spark SQL basic example") \
8     .config("spark.some.config.option", "some-value") \
9     .getOrCreate()
10
11 df1 = spark.read.format("csv").option("header", "true").load("WorldCups.csv")
12 df1.createOrReplaceTempView("t1")
13
14 q1 = spark.sql("select Winner,collect_set(Year),count(Country) from t1 GROUP BY Winner").show()
15
16
```

Setting default log level to "WARN".
To adjust logging level use `sc.setLogLevel(newLevel)`. For SparkR, use `setLogLevel(newLevel)`.

Winner	collect_set(Year)	count(Country)
Germany	[2014]	1
France	[1998]	1
Argentina	[1978, 1986]	2
Italy	[1934, 1938, 1982...]	4
Spain	[2010]	1
Uruguay	[1950, 1930]	2
Brazil	[2002, 1994, 1958...]	5
England	[1966]	1
Germany FR	[1990, 1974, 1954]	3

Process finished with exit code 0

Event Log

- 7/16/2018
- 9:40 AM IDE and Plugin U
- 9:40 AM Unregistered
- The directory C:\
- [Add root](#)
- [Config](#)

Unregistered VCS root detected: The directory C:\Users\MuthaNagaVenkataSaty is under Git, but is not registered in the Settings. // Add root Configure Ignore (today 9:40 AM)

15:1 CRLF: UTF-8: ENG 11:42 AM 7/16/2018

Type here to search